A PROTOCOL ANALYSER FOR THE MONITORING AND ANALYSIS OF OSI NETWORKS

Nasser Modiri, M.Sc., D.Phil., A.M.I.E.E., M.I.E.E.E.

EleTek Systems
1107 Second Ave, Suite 704
Redwood City, CA 94063, USA.

Abstract.

The widespread acceptance world-wide of Open Systems Interconnection (OSI) has resulted in an increasing number of computer networks being installed that are based on internationally agreed standard protocols defined by the International Standards Organisation (ISO). Associated with any computer network there must be instrumentation for the monitoring and analysis of the network performance. Most instruments currently available for use with OSI networks, however, are relatively low-level in so much that they are concerned mainly with monitoring and analysing the usage of the transmission medium. In contrast, the protocol analyser described in this paper, in addition to performing such functions, is also capable of performing detailed analysis of the interactions between any two systems connected to a network and the verification of their operation against the adopted protocol standards of the network. Also, the results of the analysis operations are presented in a high-level, graphical style that requires a minimum of knowledge about communication protocols on the part of the user.

1 INTRODUCTION

The acceptance worldwide of Open Systems Interconnection (OSI) by most large computer manufacturers and procurement agencies means that many computer networks are now being installed that are based on a standard protocol defined by the International Standards Organisation (ISO). Although prior to the commissioning of the communication subsystem within a networked computer the protocols making up the subsystem are subjected to rigourous conformance tests, should a fault be suspected in an operational system it is also necessary to provide facilities to allow the operation of a system to be monitored and any interactions anaylsed to ensure it is functioning correctly. This is particularly important during the commissioning of a new network, for example, and also when a system is being added to an operational network. This paper is concerned specifically with this aspect of OSI networks and describes the results of a research project that has been concerned with the design and implementation of a network protocol analyser that has been implemented for this purpose.

Most network analysers [1,2,3,4,5,6,7] that are currently available, simply log all network traffic - frame transmissions - in a passive way and then subsequently use the logged data to determine such statistics as the distribution of traffic levels around a network, the level of usage of each network segment, time and load related problems, verification of network error handling, and so on. Normally, a profile of the performance of the network is created and the network behaviour characteristics are then presented through a series of summary screens. The network characteristics are generally presented as bar, histogram or performance - against - time sampled graphs. Hence from the analysis, one can deduce, for example, network utilisation as a line chart with the actual data points for each sample over the sampling period. This type of statistic can be very useful to a network manager since, from the captured data, nodes, for example, that generate large quantities of data can be singled out for further analysis.

In addition to such features, some analysers provide some rudimentary facilities for investigating the actual frame interactions between two selected systems. Such analysers, therefore, are referred to as (network) protocol analysers. Normally, however, with most current protocol analysers the gathered data is presented in a basic ASCII-hex form and hence such analysers generally need expert operators who have a thorough knowledge of the protocols being analysed. In contrast, the protocol analyser to be described, provides a user-friendly facility that reduces the level of expertise required from the operator when analysing the behaviour of (OSI) networks that are based on ISO.

The paper comprises seven sections. After the introduction, Section 2 gives the rationale behind the work and the approach adopted. Section 3 describes the overall structure of some of the software tools developed. Section 4 gives some detail about how to configure and generate an analyser for a specific network environment. This is then followed in Section 5 by a description of the run-time components of a prototype analyser, and in particular, how the user can control interactively both the analysis and style of presentation of the gathered data. In the ISO standards documents, the Protocol Data Units (PDUs) exchanged between two peer protocols are in a precisely defined format and Section 5 describes how the messages gathered are decoded prior to analysis with examples of the analysis and presentation procedure adopted. Finally, Section 6 discusses some of the implementation details of the analyser in a more general context and Section 7 concludes the paper.

2 RATIONALE

The adoption of a layered architecture for the ISO reference model for OSI [8,9,10,11,12] means that each protocol layer can be treated as a separate entity for analysis and test purposes. Each layer in the ISO model provides a well defined set of user services to the layer above it and operates according to a precisely specified protocol.

Moreover, the structure (format) and contents of the PDUs exchanged between two correspondent (peer) protocol entities are also precisely defined. Typically, a PDU created by one layer will comprise its own Protocol Control Information (PCI) together with the PDU(s) from the higher layer(s) in its user data field. This is shown in diagrammatic form in Fig. 1.

The effect of this is that by analysing the composite information in each frame transmitted between two systems over a network, it is possible to deduce:

- the identity (address) of the sender and intended recipient of each frame;
- the PDUs making up the frame;
- the protocol layer to which each PDU relates;
- the protocol control information associated with each layer.

As will be expanded upon later, each frame transmitted over an OSI network consists of an integral number of octets - 8 bit bytes - and most network analysers simply present the PCI relating to each layer as a string of ASCII - hex characters. To deduce the cause of a suspected fault with such analysers, therefore, requires considerable expertise and knowledge on the part of the user.

Within the standards documents relating to the ISO protocols well understood techniques have been adopted for defining both the user services provided by a protocol layer and also the specification of the operation of a protocol entity; the former is in the form of a time sequence diagram whilst the latter is normally in the form of an event-state table.

Clearly, therefore, if when analysing the operation of either a single protocol layer or a complete communication subsystem the various PDUs and associated events can be related directly to the relevant graphical template, then it will be more straight forward for a user of the analyser firstly, to follow (and hence understand) the sequence of events which have occurred in relation to the standards document and secondly, to determine the cause of any errors should these arise.

When carrying out a particular test run, the amount of data gathered can be considerable. Thus, when presenting the results of a test run it is important that this is done in an interactive way so that the user can firstly, proceed with the analysis operation at his or her own rate and secondly, dictate the sequence and amount of detail which is to be displayed. For example, if a user is interested only in, say, the Transport Layer [13,14], then it is important to display only the messages which relate to it. Alternatively, if a user is interested in the behaviour of a complete communication subsystem, then the interactions and PDUs relating to all the layers must be presented. In either case, however, the presentation of this information should be in a top-down manner so that the level of detail displayed is under the control of the user. This is the approach which has been adopted for the design of the analyser to be described.

3 OVERALL STRUCTURE

Before describing the detail features of the prototype network protocol analyser that has been developed [15,16], it is perhaps helpful to first identify the major software components that make up the analyser. A schematic of the various components is shown in Fig. 2.

Essentially, there are two major software components - one concerned with Environment Selection and the other Run-time components. The function of the Environment Selection component is to allow the user to tailor the analyser to the requirements of a particular open system environment. This is a one-off operation and, once carried out, the analyser is ready to be connected to the network.

The Run-time component is itself comprised of three sections - *Monitoring*, *Message Decode* and *Analysis*. The *Monitoring* component is concerned with the capture of messages from the network medium. Clearly this operates in real-time and the captured messages are then stored on disc to await further processing. The *Message Decode* component operates on the captured data and is responsible for parsing each logged message to identify the individual PDUs that make up the message. Once this has been carried out, the *Analysis* software is then run. The latter provides the user with an interactive facility to allow the style of presentation of the analysed results to be selected. Each of the major components will now be described.

4 ENVIRONMENT SELECTION

It should be remembered, there is not just a single protocol standard for each layer in the ISO reference model. Rather there are a number of standards each offering a different level of functionality within the context of the layer to which it relates. Moreover, as experience is gained from the use of a particular standard, so revisions are made to it, often with significant differences from the original version(s). Clearly, therefore, it is important when creating an open system environment for the administrative authority to specify not only the particular standards to be used at each layer but also the version numbers of these standards. The resulting set of standards is then referred to as the configuration profile or road map of the environment. When selecting a network protocol analyser, therefore, it is necessary to ensure that it has the correct configuration profile for the network environment that is to be used. An example, of an environment that is based on the ISO protocols is MAP [17]. There is now a fully defined configuration profile for this known as MAP3.0 and indeed it is for this environment that the network protocol analyser to be described has been initially targeted.

4.1 Configuration profile

The set of protocols to be used with the analyser are selected from a list of possible protocols using an interactive software tool known as *Configure*. An example of its use is shown in Fig. 2 (a..h).

The example relates to the selection of a configuration profile for MAP3.0. The operator first selects the appropriate protocol suite needed to make up an analyser for use in this environment. It should be noted that although the selections in the example are specific to the MAP3.0 analyser, the software is designed to be able to handle any combination of the standards specified in the menu. As can be seen, as each layer in the reference model is selected then all the protocol standards supported by the analyser for the selected layer are listed. The user is thus able to select the appropriate protocol to be used for each layer simply by using a selection button on the mouse. Then as each protocol is selected, this information is added to the configuration file of the analyser. The latter is included with the parsing code for each layer prior to compilation. Hence, after compilation, just the code necessary to parse the selected protocols will be produced.

4.2 PDU template generation

Once the set of protocols to be used in the environment have been selected, it is then necessary to generate a precise definition of the PDUs associated with each selected protocol.

Many of the ISO standards have a number of optional features associated with them; for example, with respect to the inclusion of a particular field within a PDU. Hence, having selected a configuration profile, it is then necessary for the user to select the inclusion (or otherwise) of those fields that are optional in each protocol making up the selected profile. This is accomplished using the interactive software tool *Generate* and an example of its use is shown in Fig. 4 (a..d).

As can be seen, as each protocol layer is selected, the list of PDUs associated with the selected protocol are listed. The user then selects each PDU in turn and, if any of the fields are optional, the user is prompted to specify whether the field should be included or not. Thus after this phase a precise template for each protocol in the suite will have been generated and this is used in the Run-time phase by the corresponding parser code generated as a result of the earlier configuration operation.

5 RUN-TIME COMPONENTS

Before describing the features of each of the Run-time components in detail, it is perhaps helpful first to outline a typical user dialogue with the implemented system. A schematic representing the overall menu is shown in Fig. 5.

As with all interactive systems, the menu is hierarchical in structure with the user selecting a specific option at each level in the hierarchy. Thus the highest (system) level menu allows the user to enter the required operational mode: *Monitoring*, *Message Decode* or *Analysis*.

Normally, the *Monitoring* mode is selected first as this is concerned with the logging of the frames transmitted on the transmission medium. The user will then select the *Message Decode* mode and this will use the previously created layer parsing code (engine), together with the PDU templates where appropriate, to deduce the individual PDUs PCI making up each logged message frame. Finally, the user selects the *Analysis* mode to present the results of the decoded message (frame) sequence to confirm or otherwise the correct operation of the targeted system.

5.1 Monitoring

The *Monitoring* mode allows the messages exchanged during a dialogue between two (or more) selected systems to be monitored and logged. The user first enters the (network) addresses (or names) of the two systems and these are first checked against a file - the system directory - that contains a list of all the names and corresponding addresses that are in use. The network addresses (or names) are then used as a filter to log only those messages which are exchanged between these two systems. Fig. 6 part (a) shows the operator has selected File Transfer Access and Management (FTAM)[12] as the application entity of interest and part (b) shows the operator entering the network addresses of the two nodes of interest.

The operator is then requested to specify the number of messages that it requires to capture during the *Monitoring* phase. Once the selected number of messages have been captured and stored on disk (or a timeout interval expires), the network protocol analyser is then ready to analyse them. This is achieved; firstly by parsing each message to ascertain the PDUs it contains - *Message Decode* - and then analysing the content and sequence of the PDUs to verify the correct operation of each system - *Analysis*.

5.2 Message Decode

The parsing code for each protocol in the selected configuration (produced by the *Configure* program) is known as a parser-engine. Also, associated with each parser-engine, there is a script file that manages the output of the parser-engine. There is then an additional script file which invokes each parser-engine in the correct sequence and also manages any data produced by them.

Essentially, as each parser-engine is called, it takes in a Presentation Stream (PS) of hexadecimal digits - the contents of a logged frame - parses it, and then displays the parsed contents in a readily understood form. Alternatively, if an error is detected when parsing the PS, then the operator is warned and a message indicating where and which octet(s) of the PS is(are) in fault is written against the faulty octet(s). Although all the PCIs relating to each protocol are parsed, only the essential fields in the PCI are printed. In addition, however, it is possible to request a more complete version of the PCI should this be required.

As an example of the *Message Decode* phase, assume that it is required to analyse the behaviour of two communication systems that are exchanging messages over a factory-wide broadband MAP network. A typical sequence of messages exchanged (in ASCII-hex form) with such a network are as shown in Fig. 7.

The message sequence relates to two FTAM protocol entities and these have been obtained by simply monitoring all transmissions over the factory-wide broadband transmission medium and using the physical network addresses of the two systems as a filter.

5.3 Analysis

As was mentioned earlier, because the amount of information gathered during a monitored period may be large, it is important to provide a facility to enable the user to interactively select, in a top-down way, the level of detail relating to the interactions which he/she is interested in. Thus, as can be seen in the menu shown earlier in Fig. 5, when the *Analysis* mode is selected, the user is asked to select the level of interactions which are of interest - System Level, Layer Level or Logical Sequence Checker (LSC). An example of each will now be presented. Each uses the sequence of messages shown earlier in Fig. 7.

System Level

The seven-layers in the ISO reference model can be divided into two groups according to their function - application-oriented (which embraces layers 5-7) and network-oriented (which embraces layers 1-4). Thus, when the System Level option is selected, the user is then asked to select the required functional group. As an example, Fig. 12 part (a) shows the user selecting the application-oriented layers and the interactions shown in part (b) and part (c) of this figure relate to the application-oriented group.

Essentially, the decoded message sequence is first preprocessed and only those messages which have PDUs relating to the application-oriented layers are analysed. As can be seen, the interactions are shown against a graphical template and the user steps through the interactions which have taken place (deduced by analysing the messages exchanged) simply by pressing a key on the mouse. Also, the lower analysis window acts as a commentary window and gives useful comments about the analysis process being carried out within each parsed frame. In addition, a vertical bubble bar is provided on the side of this window, the height of the bar represents the position of the displayed frame within all the messages captured.

A similar procedure is followed to display the interactions relating to the network-oriented group and an example sequence is shown in Fig. 13 part (a) and part (b).

A similar preprocessing phase is carried out but this time only messages which contain PDUs relating to the application-oriented layers analysed. Also, as can be seen, a different graphical template is used but, apart from this, exactly the same procedure is followed.

Clearly, if during the displayed sequence a fault is suspected, the user will then require additional or more detailed information to try to ascertain the cause of the fault. Normally, this will be related to a specific protocol layer and hence the Layer Level would then be selected with the suspected faulty layer as the target. In this way, the user is able to obtain first a high level view of the operation of the system then, should a fault be suspected, a more detailed view of the suspected area.

Layer Level

To analyse the interactions involving a specific layer, the user selects the Layer Level option and, as an example of its use, part (a) of Fig. 14 shows the user selecting the Layer Level option and part (b) shows the type of graphical template used.

The layer (protocol entity) selected is FTAM and the steps shown again relate to the message sequence in Fig. 7. For this type of analysis the messages are again preprocessed and only those which have PDUs relating to the selected layer are analysed. The style of presentation is intended to show the services provided by the layer and hence the graphical template used is a time sequence diagram.

As with the System Level display, the user also may select additional information if this is required in the lower analysis window. A scrollbar is also provided on the side of the window so that the user can go back to the previous PDU being displayed. Also, information is provided in the window to indicate which frame in a sequence is under analysis. Thus, in the example shown in part (c) of Fig. 14, the user has selected to display more information relating to the INIRQ FTAM PDU. As was described earlier, the various FTAM PDUs are defined in ASN.1 form and hence, since this definition is required (for each PDU type) for the decoding process outlined earlier, the various fields associated with the PDU are displayed against the corresponding ASN.1 template to aid readability. Clearly, if a complete sequence of messages is being analysed, in addition to analysing - and displaying - selected messages, it is possible to relate the sequence to the formal definition of the protocol entity of the layer under investigation. By analysing a sequence of frames, it is possible to ascertain the correct (or otherwise) functioning of the protocol behaviour.

The analyser must respond to any type of combination of PDUs which have been transmitted between the two nodes under consideration. For this reason, all the graphical templates devised can react dynamically to any combination of decoded PDUs. In particular, when preparing the graphical templates care has been taken to make them suitable to respond to the dynamic behaviour of a test run. The display interactions may start at any position in a test. Also, should a frame be encountered that is faulty (that is, has not been decoded), then the user is alerted of this.

As was depicted earlier in Fig. 2, each message contains PCI relating to a number of protocol layers and hence the first step is to decode each message into its constituent parts based on the known structure of each PDU as defined in the relevant standards document. The decoded PDU boundaries relating to the messages shown in Fig. 7 are thus as shown in Fig. 8 part (a) and part (b).

As can be seen, the first two messages comprise only PDUs up to the Transport Layer whilst the others comprise PDUs for the full seven-layers. In practice, this is readily determined since, after each parser-engine has been called - starting at the MAC Sublayer - a test is made to determine whether there is any data remaining in the frame buffers; if there is the next parser-engine is called, otherwise the parsing is complete.

To illustrate the above procedure, consider the first message block in the sequence. The various steps involved are shown in Fig. 9. First the protocol control information relating to the MAC Sublayer is determined by using the MAC parser-engine derived from the ISO 8802.4 standard. Similarly, the PCI relating to the LLC Sublayer is determined (again using the appropriate parser-engine derived from the LLC ISO 8802.2 standard document) and so on until the complete message has been parsed.

In the standards documents the structure and contents of each PDU up to and including the Session Layer [19,20] is defined in a fixed bit/octet string form. The PDUs associated with the Presentation Layer protocol machine [21,22]and the various application layer entities, however, are defined in an Abstract Syntax Notation (ASN.1)[23,24,25,26]. As an example, a PDU relating to the FTAM protocol is shown in part (a) of Fig. 10 and part (b..e) outlines how the related PDU is encoded.

As can be seen, each field (data element) in the PDU is of an abstract data type and the corresponding value making up the PDU is then encoded as:

- Type - this indicates the type of the data element; both simple types such as INTEGER, BOOLEAN etc. are used and also constructed types such as SEQUENCE (similar to a Record type) and SET;
- Length - this indicates the number of octets which follow in the contents field;
- Contents - this is the actual value of the data element, represented in a fixed (concrete) syntax form if it is a simple type, or a further type field if it is a constructed type.

Each PDU will have a different encoding (and hence decoding) procedure associated with it but, since the first octet always defines the type of the PDU, then the appropriate procedure can be selected and used to interpret (decode) the values associated with each data element. As an example, Fig. 11 shows the output of the parser-engines for the application support layers including the Session Layer.

Once the monitored and logged message sequence has been decoded - first into individual PDUs and then the various fields (elements) in each PDU identified - it can then be analysed and displayed to enable the user to observe the interactions that have taken place. Basically, in this phase a long file is generated relating to all the message frames parsed by the *Message Decode* and the output of the parser includes enough information in the generated file to determine if a PDU is in error.

When a PDU is detected to be in error, an audible and a visible signal are given to the user in the form of sounding the keyboard bell and flashing the screen, plus putting the erroneous PDU name in inversed screen mode.

A PDU in error is determined either by its Unique Identifier (UI) not being recognised as being a PDU belonging to the layer or while parsing the individual fields within a PDU an out of range field is detected.

There is also a help facility provided for searching through any of the parsed (decoded) files produced by the parser-engines in their appropriate windows. These windows may be invoked individually or in any combination. The contents of the parsed file, therefore, may be printed and used to identify any errors that may be present prior to selecting the *Analysis* mode. Alternatively, if there are a large number of frames being decoded, the parsed file may be large and a more user-friendly analysis option is thus offered to the user.

Operational modes

There are two alternative *Message Decode* processes and hence operational modes - Manual and Automatic. With the Manual mode of operation, the user has to specify the frame number to be decoded. Consequently, this has a slower performance rate, but a number of help facilities are provided at each stage of parsing the individual PDUs making up the frame. For example,the user can enquire about the parsed PDUs for the same layer in other frames in the sequence by simply paging the appropriate window. If at any time an error is encountered within a PDU being parsed the audible and visible bells are invoked. The biggest advantage of the Manual mode is that the parsing of frames does not have to be done in the same order as the stored sequence of frames. This mode is intended, therefore, as a means of examining the contents of individual frames rather than the sequencing of frames.

With the Automatic mode of operation, the user may elect to have the total parsing operation (that is, of all the logged frames) done automatically without the display of any of the parsed output. In addition, the user may elect to specify, say, a specific layer protocol in which case only the parsed output relating to that layer will be produced. In either event, however, the output of the Automatic mode process forms the input of the *Analysis* process.

Logical Sequence Checker (LSC)

A second type of analysis that can be done on the parsed PDUs is semantic analysis of the PDUs. This involves checking the sequence of PDUs and hence only when the parser-engines have syntactically checked the contents of all the frames in a logged sequence this is carried out.

The checking process involves a thorough knowledge of the interrelationships of the PDUs for a layer. This knowledge is based on the specification of the state-machine of the selected protocol of the layer under examination. All the possible valid sequences of PDUs relating to the operation of a protocol machine are stored in a table. This information is then used by the LSC to determine with a good probability, whether the sequence of the PDUs exchanged between two communicating protocol entities is in a valid sequence.

As an example, the contents of the sequence table for the FTAM protocol entity is shown in Fig. 15. This has been constructed by analysing the operation of the FTAM state-machine given in the ISO/DIS 8571/4 FTAM specification Document: '*' indicates a valid sequence and a blank an invalid sequence. As can be see, a protocol such as FTAM has 41 PDUs associated with it and hence manual sequence checking is difficult. The LSC thus takes much of the burdon away from the operator.

The selectable options available with the LSC provides the operator with the capability of checking for the correct sequence of the PDUs in the logged parsed messages for all the protocol entities supported by the analyser in any of the seven layers. The input file used by the LSC is again the output file produced by the *Message Decode* process. An example of the output of the LSC is shown in Fig. 16.

This again relates to the logged message (frame) sequence shown earlier in Fig. 7. As can be seen, the selected protocol layer under test is transport and, as each TPDU in the sequence is processed, an indication is given wether the PDU is valid or not (i.e., is or is not in the correct sequence). The algorithm of the LSC is based on a two dimentional table driven procedure. The incoming PDU under test is then used as one index and the last PDU processed as the other index.

6 DEVELOPMENT ENVIRONMENT AND TESTING

The network protocol analyser described has been implemented on a Sun Workstation® [27]. The programming language used is C [28] running under the UNIX™ operating system. The source code of the basic analyser - comprising the code necessary to analyse the MAC through Presentation Layers - occupies 1 Mbyte. Additional code is then required for each application entity supported; for example, FTAM requires 80 Kbytes and MMS [29,30] 100 Kbytes. A typical executable version of the analyser requires in the order of 3 Mbytes of main memory which includes the additional graphics and window software tools for the man-machine interface discussed.

The menu facility used with the analyser operates in the Suntools environment. SunView (Sun Visual/Integrated Environment for Workstations) graphic primitives are used for devising the interactive graphic analysis of the analyser. Only the SunView graphic package has been used to devise the graphics utilities. Hence, if there is a requirement for changing the graphic primitives to that of another package, changes can be made easily. There are three graphic programs associated with the analyser: *Schematic*, *Network Statistics* and *Analysis*. Since they all use the Sunview graphic primitives, they have been merged into a single program. An argument is then passed to the executable merged program, so that individual programs may be called independently from one another.

Testing

The physical interface to the analyser is, of course, network dependent. Essentially, however, it is through a communication card containing just the appropriate physical interface circuitry for the network being used. The filtering and message storage operations are then performed in real-time. Also, in addition to performing the filtering operation, statistics relating to network usage are computed and maintained.

Two examples of communication cards for use with the Sun are the MVME374 Ethernet™ Controller Module [31] and the MVME372 Advanced MAP Controller. The first provides a Sun interface to an 802.3 Ethernet™ network and the second an interface to an 802.4 token-bus network. Both these boards contain a separate processor and local memory that is used for the corresponding protocol software: the first the complete TOP [32] software suite and the second the complete MAP suite. With the analyser, however, the latter is not used, of course, and instead the physical network interface controller operates in the promiscuous mode and a small program is then used that simply performs the filtering and statistics gathering operations.

Both boards can monitor the cable in real-time and deduce frame exchanges between two specified systems. The frame exchanges between the two specified systems are passed to the host system for processing. The frames are time stamped and stored on disc to await the decode and analysis processing. In addition, the frame contents are printed on one of two windows: one to display all the frames sent by the source node and the other to display all the frames sent by the destination node. In this way, the operator can readily see that the system is performing the *Monitoring* operation and, hopefully, both systems are exchanging frames.

For test and development of the analyser two different approaches were used. The first used a set of programs (one per layer) that were written to allow a user to interactively create composite frames comprising any selection of layer PDUs. A set of such frames were then used for the earlier test and development phases.

In the second approach the analyser was tested with real data gathered from the communication between two Sun Workstations® running the MAP3.0 software. A subset of these frames was shown earlier in Fig. 7 and all the analysis operations were performed on this data to verify its correct operation.

7 CONCLUSIONS

The aim of the work described in this paper has been to develop a suite of software tools to aid the detailed analysis of the interactions between two computers connected to an OSI network that is based on the ISO protocols. The results of the analysis operation are presented to the user against suitable graphical templates. Also, the level of detail in the presentation of the results can be interactively controlled. The result is a high-level, user friendly interface that requires a minimum of knowledge of protocols on the part of the operator.

In addition to being used with the analyser, some of the software tools described are also being used as the basis of other instruments. For example, a Dynamic Tester [33] is currently being implemented that allows a suspected faulty system to be tested dynamically and the cause of the fault diagnosed automatically. This employs the same Environment Selection software and also the same *Monitoring* and *Message Decode* Run-Time components. Selected test scenarios each comprising composite frame sequences are first created, together with the expected response frames, using these tools. The actual and expected response frames are then compared and any differences highlighted against the corresponding PDU templates.

8 ACKNOWLEDGEMENTS

The work reported in this paper has been partially funded by The Networking Centre Ltd. of Hemel Hempstead, England. The author would like to acknowledge this support and also the many usefull discussions with them about the facility affered by the analyser.

9 REFERENCES

[1] EXCELAN: 'Excelan Networking', IEEE Spectrum, 1988.
[2] NETWORKII.5: 'NetworkII.5', IEEE Spectrum, 1987.
[3] Spider Systems: 'SPIDERMONITOR, Ethernet Monitor', Spider Systems Ltd, 1987.
[4] Hewlett Packard: 'HP 4951C and HP 4952A Protocol Analysers', Hewlett Packard Inc., 1985.
[5] Hewlett Packard: 'X.25 Network Performance Analyser for the HP 4953A', Hewlett Packard Inc., 1985.
[6] Trend Data Link: 'INTERVIEW 7000 Series Protocol Analysers', Atlantic Research Corporation, 1987.
[7] Network General: 'The Sniffer, The Case of 'Slower is Faster'', Network General Corporation, 1987.
[8] ISO 7498 (1988), OSI: Basic ReferenceModel for OSI.
[9] ISO 7498-1 PDAD 2 (1988), Part 1,Multipeer Transmission.
[10] ISO/DIS 7498-2 (1988), OSI: Final Text ofPart 2, Security Architecture.
[11] ISO/DIS 7498-3 (1987), Part 3, Naming and Addressing.
[12] ISO/DIS 7498-4 (1988), Part 4, Management Framework .
[13] ISO 8072 (1987), OSI: Transport Service Definition.
[14] ISO 8073 (1987), OSI: Transport Protocol Specification.
[15] Modiri, N.: 'Graphical Presentation of Protocol Behaviour', The University of Sussex, Internal Report, 1987.
[16] Modiri, N., Halsall F.: 'An Analyser for Monitoring and Analysis of OSI Protocol', NETWORKS '88, 1988.
[17] General Motors Corporation: 'Manufacturing Automation Protocol', Manufacturing Engineering and Development, Advanced Product and Manufacturing Engineering Staff (APMES), GM Technical Centre. 1988.
[18] OSI: File Transfer, Access and Management, Final Text:
DIS 8571-1 (1988), Part 1, General Introduction.
DIS 8571-2 (1988), Part 2, Virtual Filestore Definition.
DIS 8571-1 (1988), Part 3, File Service Definition.
DIS 8571-4 (1988), Part 4, File Protocol Specification.
[19] ISO 8326 (1987), OSI: Basic Connection Oriented Session Service Definition.
[20] ISO 8326 (1987), OSI: Basic Connection Oriented Session Protocol/Specification.
[21] ISO 8822 (1987), OSI: Connection-Oriented Presentation Service Definition.
[22] ISO 8823 (1987), Connection-Oriented Presentation Protocol Specification.
[23] ISO 8824 (1987), OSI: Specification of Abstract Syntax Notation One (ASN.1).
[24] ISO 8824 PDAD 1 (1987): ASN.1 extensions to ASN.1

[25] ISO 8825 (1987), OSI: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).
[26] ISO 8825 PDAD 1 (1987): ASN.1 extensions to ASN.1 Basic Encoding Rules
[27] England D.: 'A Sun User's Guide', Macmillan Education Ltd., 1987.
[28] Kernighan, B.W., Ritchie, D.M.: 'The C Programming Language', Prentice-Hall, 1987.
[29] ISO/DIS 9506/1: Manufacturing Message Specification - Service Definition.
[30] ISO/DIS 9506/2: Manufacturing Message Specification - Protocol Specification.
[31] Motorola: 'Motorola and VMEbus' Motorola Computersysteme, 1987.
[32] Farawich, S.A.: 'Communicating in the Technical Office', IEEE Spectrum, 1986.
[33] Halsall, F., Modiri, N.: 'A Dynamic Tester For Use With OSI Networks', Submitted to Telecommunication for Publications, May, 1989.

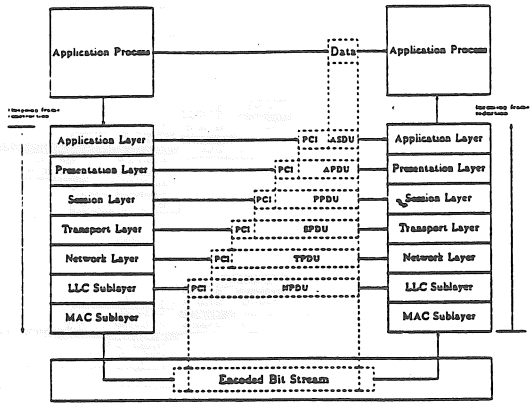


Figure 1: *Multiple Layer Interactions.*

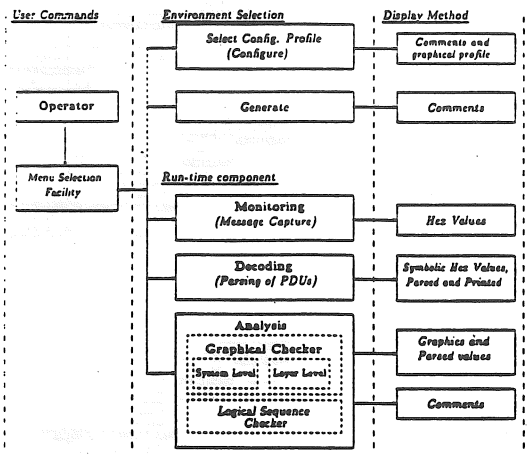


Figure 2: *Functional Representation of the Network Protocol Analyser.*

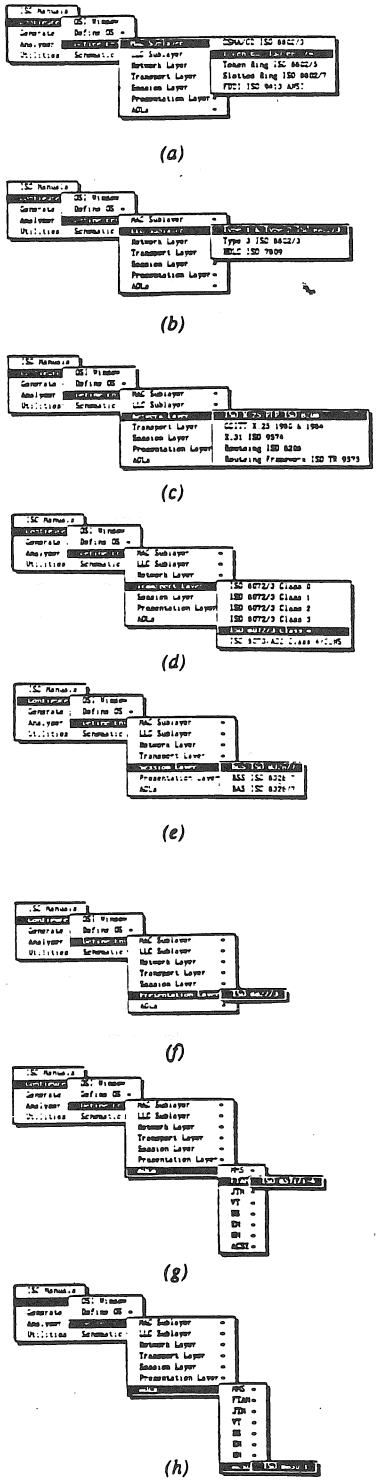
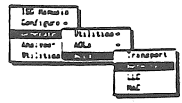
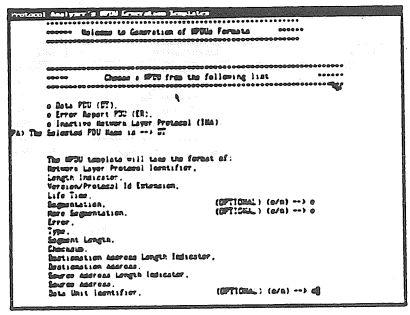


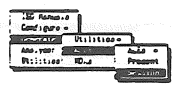
Figure 3: Configuration Profile: (a) MAC Sublayer; (b) LLC Sublayer; (c) Network Layer; (d) Transport Layer; (e) Session Layer; (f) Presentation Layer; (g) ACSE Application Entity; (h) FTAM Application Entity.



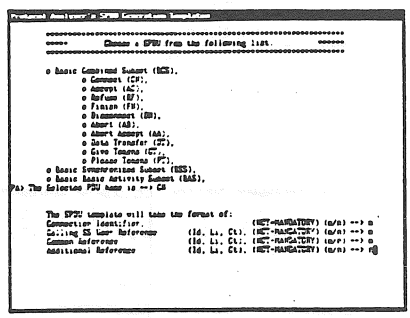
(a)



(b)



(c)



(d)

Figure 4: PDU Template Generation: (a) Select Network Layer; (b) Define DT NPDU; (c) Select Session Layer;(d) Define CN SPDU.

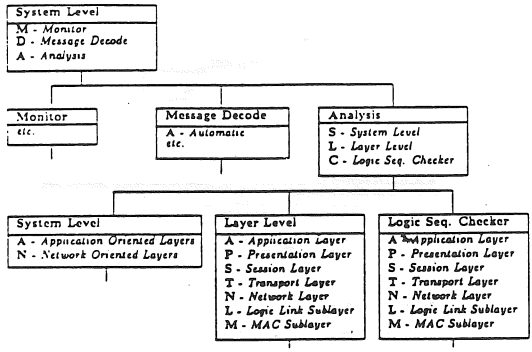


Figure 5: *Hierarchical Menu of the Network Protocol Analyser.*

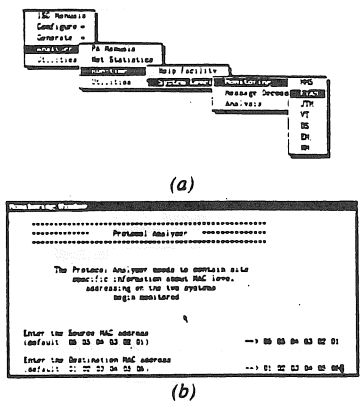


Figure 6: Monitoring Phase: (a) Select FTAM Application Entity; (b) Specify Source and Destination Address.

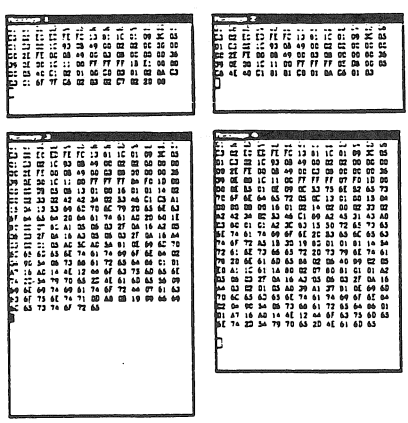
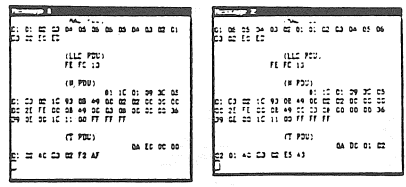
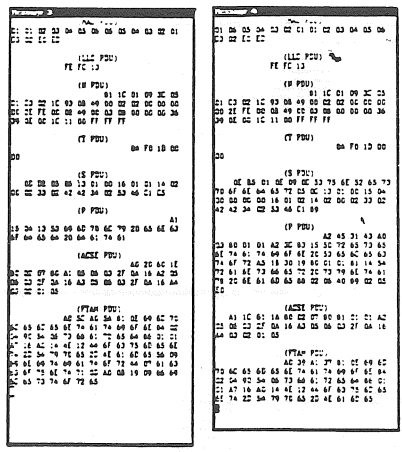


Figure 7: *Typical Message Sequence.*



(a)



(b)

Figure 8: *PDU Boundaries: (a) Transport Layer Frames; (b) FTAM Application Entity.*

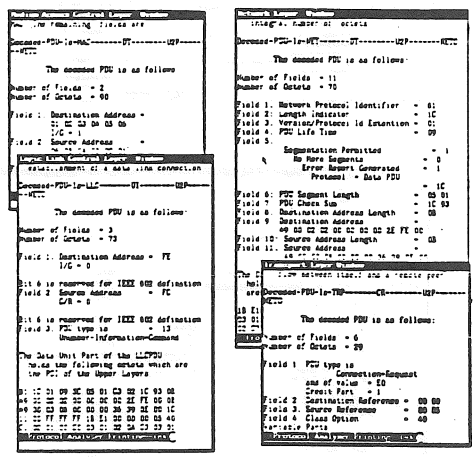
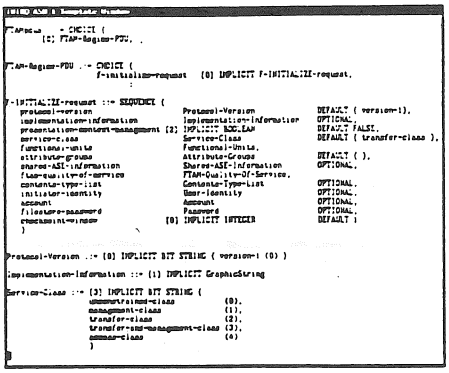
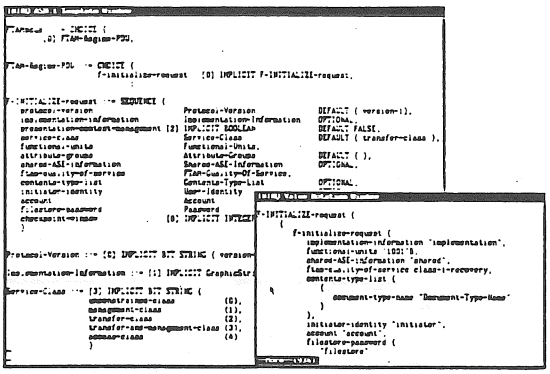


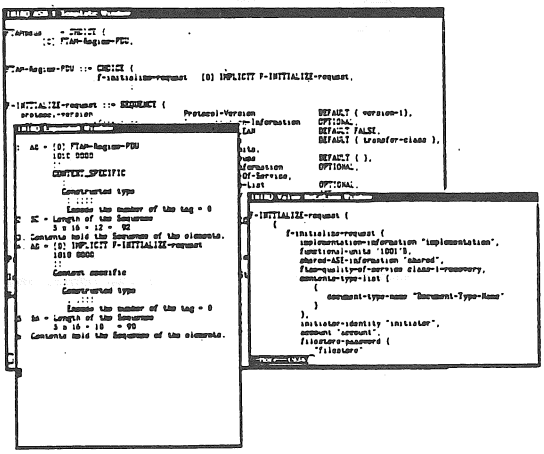
Figure 9: ***First Message Block Parsed.***



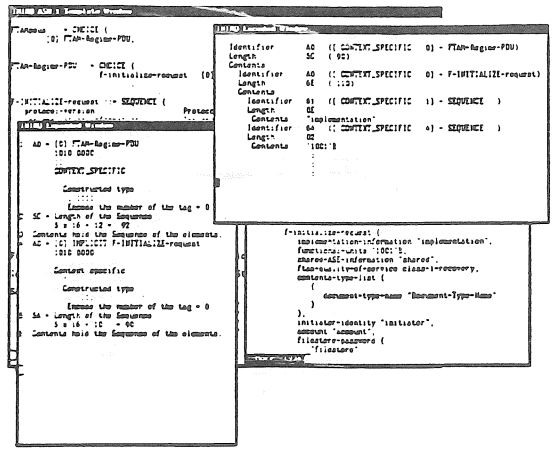
(a)



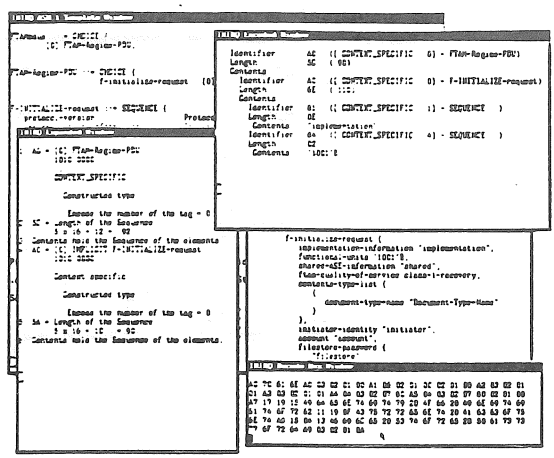
(b)



(c)



(d)



(e)

Figure 10: Encoding FTAM INIRQ PDU: (a) INIRQ PDU; (b) Concrete Value Notation; (c) Encoding Procedure; (d) INIRQ contents; (e) Concrete Syntax:

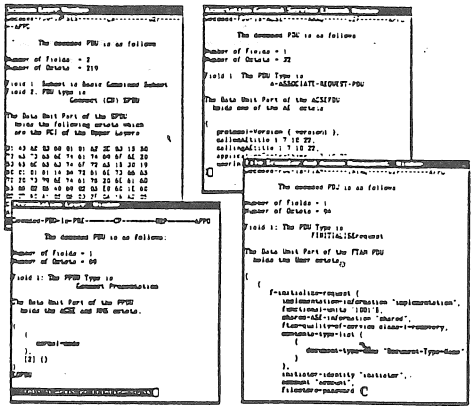
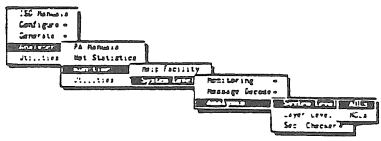
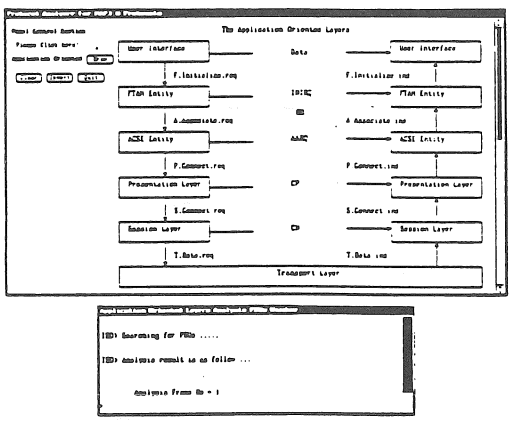


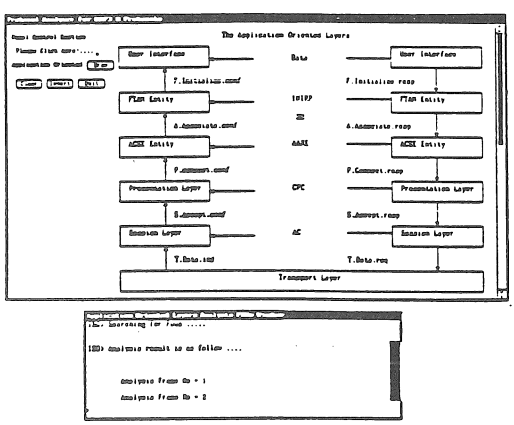
Figure 11: *Second Message Block Parsed.*



(a)

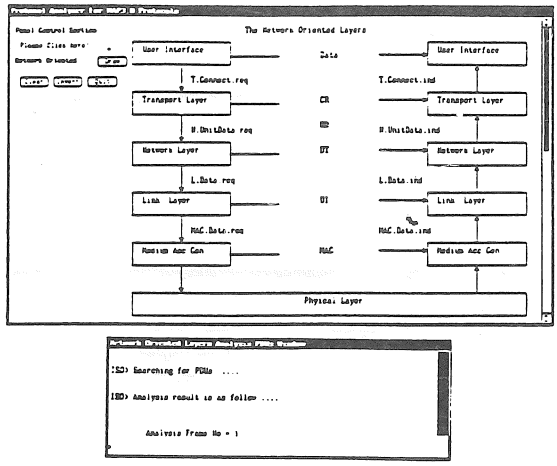


(b)

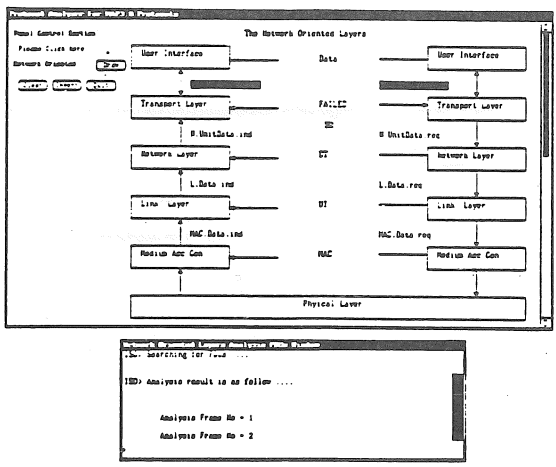


(c)

Figure 12: Analysis of the Aplication Oriented Layers: (a) Select FTAM Application Entity; (b) Initialising Frame; (c) Responding Frame.

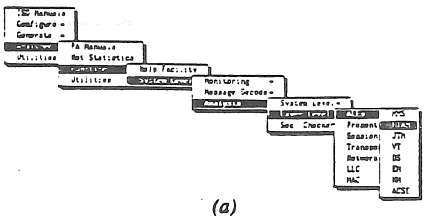


(a)

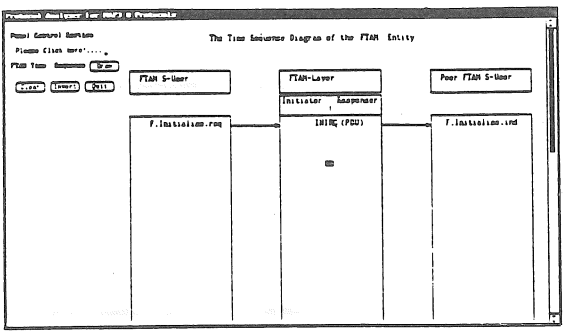


(b)

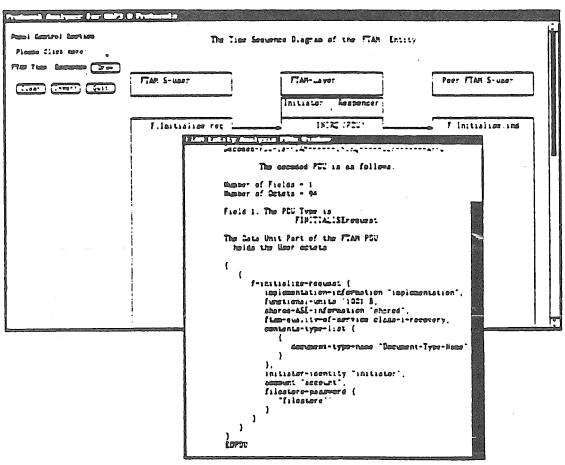
Figure 13: Analysis of the Network Oriented Layers: (a) Connection Request Frame; (b) Connection Confirm Frame.



(a)



(b)



(c)

Figure 14: Layer Level:(a) Select FTAM Application Entity; (b) Time Sequencing Diagram; (c) The Parsed INIRQ FTAM PDU.

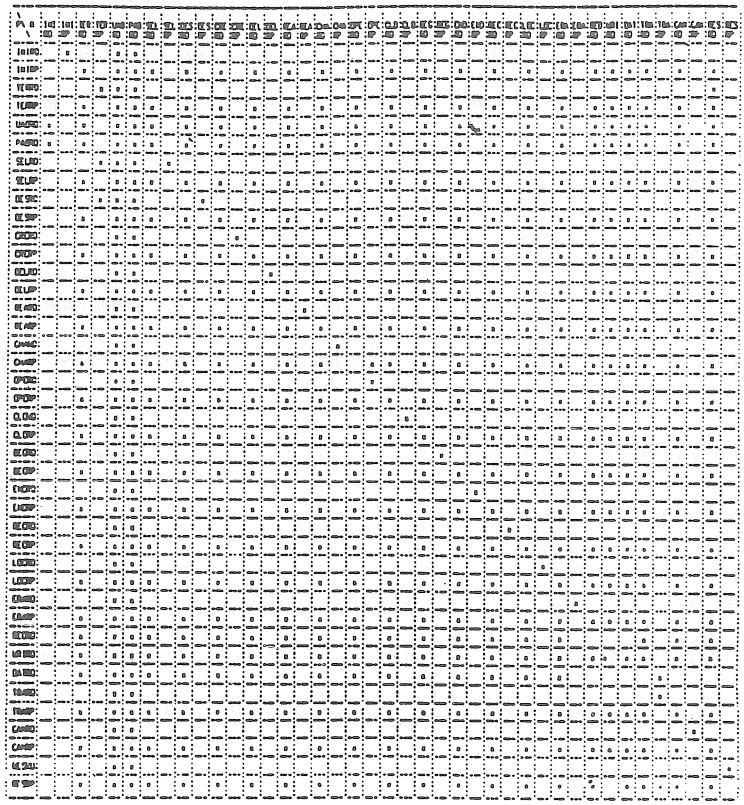


Figure 15. *Sequence of the FTAM PDU.*

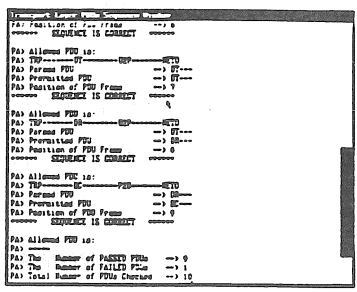


Figure 16: *LSC of the Transport Layer PDUs.*